

The Software Engineering of Applications

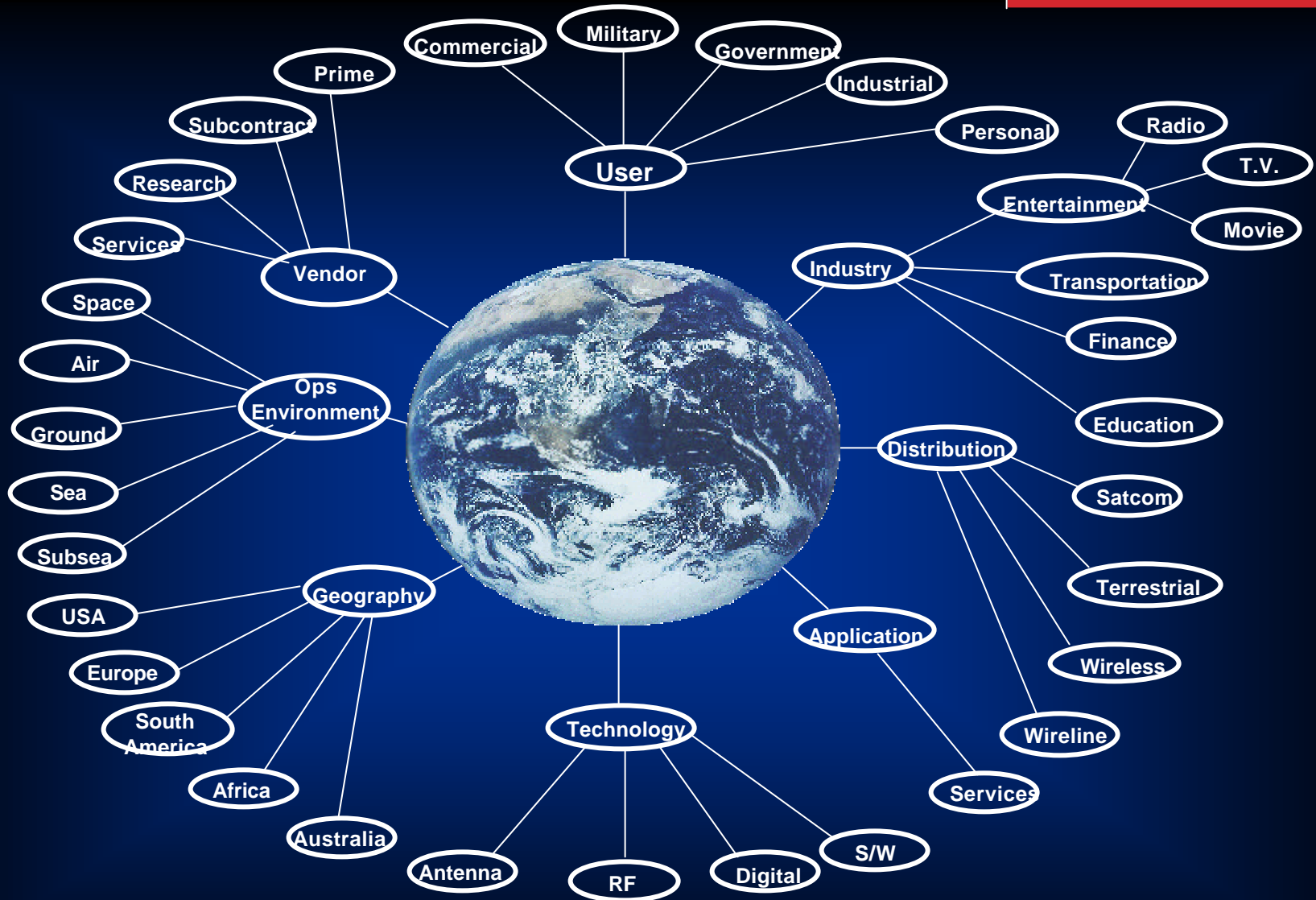
Deploying Mission Critical Applications in an Era of Pervasive Computing

2003 Symposium on Applied Computing

Rick Simonian

rsimonia@harris.com

Infocentric Planet



- Everything is Going Digital
 - Computers 1946
 - Letters 1960s
 - Slide Rules 1972
 - Watches 1973
 - Pinball 1976
 - Telephone Backbone 1980
 - Music 1982
 - Maps 1994
 - Television 1994
 - Video 1995
 - Cinema 2000
 - Radio 2002
 - People 2035?

4 Eras of Software Industry



Networked
Computing

Desktop Computers,
Dominant designs,
Packaged software

ISVs, Unbundled
Custom Software

Development &
commercialization of
the computer. Standard
Software architectures

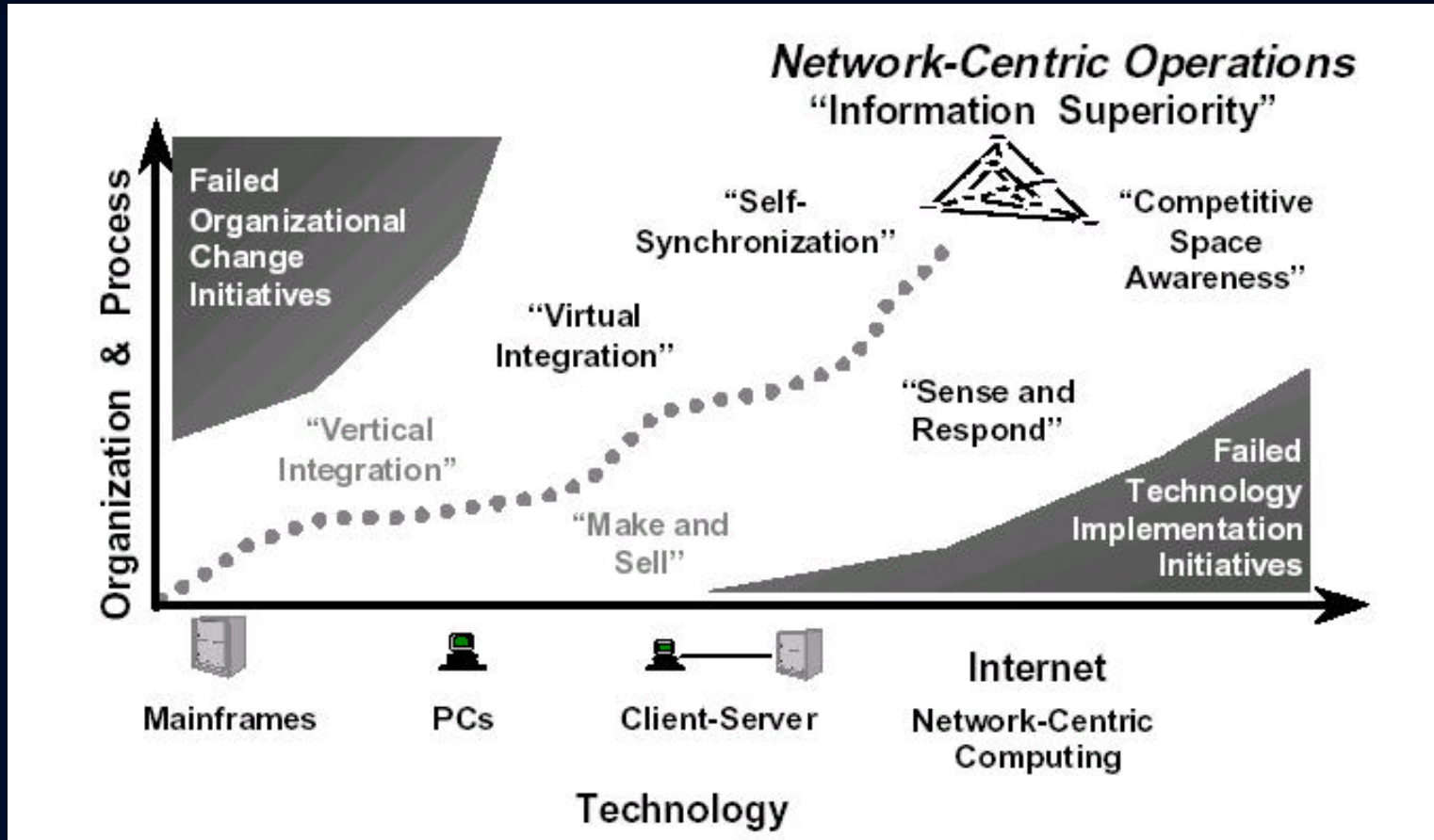
1945

1965

1978

1993

2003



Network Centric Warfare, 2nd ed., D. Alberts, J. Garktko, F. Stein, 1999

Tomorrow's Competitive Advantages



Industrial Age
Transportation Age
Chemical Age

Information Age

Biologic,
Genetic Age

Exploited Innovation

Niche Products & New Processes

Lean Mfg Speed

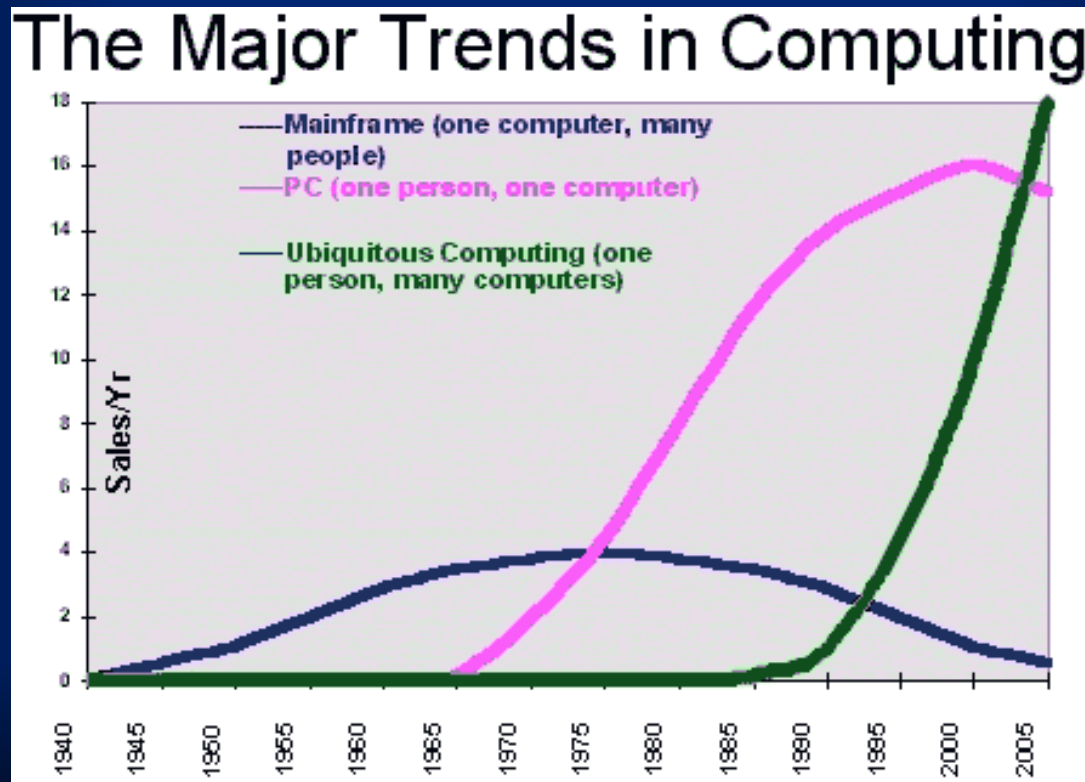
Quality & Cost

Global Scale

Prior 1970 1980 1990 2000 2010 2020

“The convergence of nanotech, biotech, and information technology will drive the future of R&D.”
Rita Colwell
NSF Director
October 2002

What is this word, “ubiquitous”?
I see it everywhere!



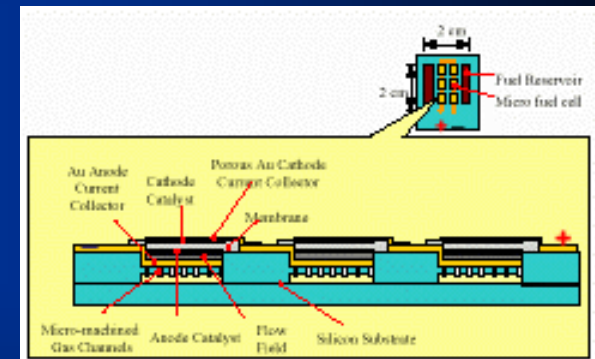


Sensors Everywhere

Grid Computing

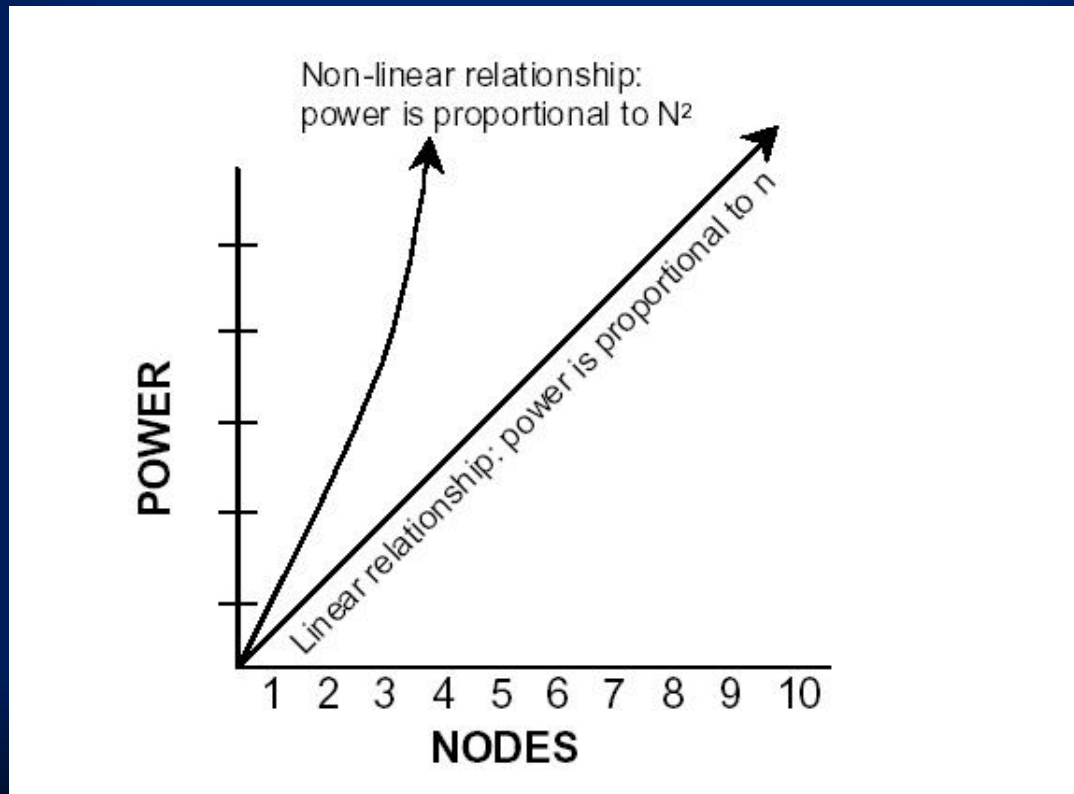
“We’re moving into a future in which the location of [computational] resources doesn’t really matter,” Ian Foster, Argonne National Laboratory

- MEMS and NEMS
- Circuits and batteries that can be printed onto cloth or paper
- Miniature power sources, such as fuel cells, jet engines, Wankel engines



- All these advances will help pervasive computing “disappear” into our environment

A digital device by itself might be useful,
But a connected digital device can
change behavior



Metcalfe's
Law

- Research in virtually every area is dependent on computation
 - Physics is almost entirely computational
 - Hardware design is starting to look like software (VHDL, ASIC design)
 - Biology depends on computer simulation
 - Computational chemistry, computational neuroscience, computational genetics, computational immunology and computational molecular biology
 - Increasing usage in sociology and anthropology
 - Generating Celera's computerized genomic map required scrutinizing over 80 terabytes

*NY Times, 3/25/01

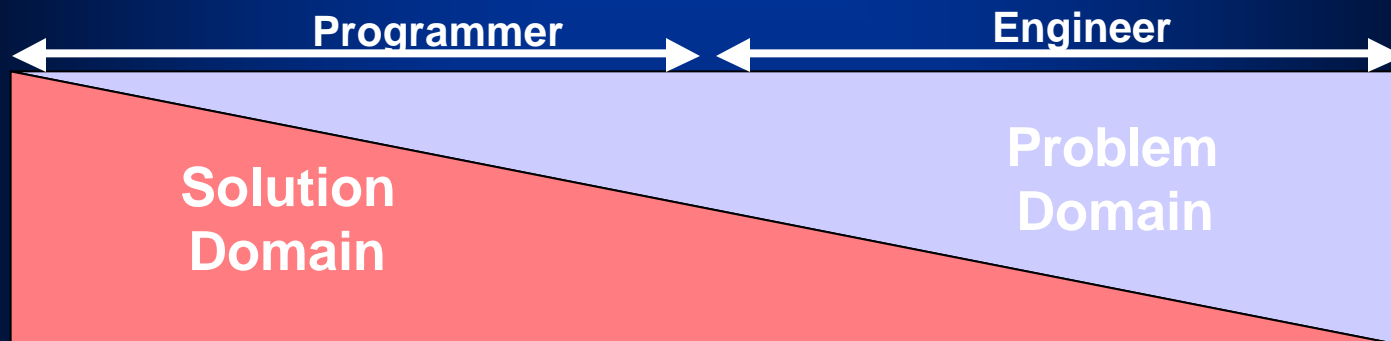
The very nature of computing is changing!

- Traditional Silicon
- Cells (e.g., your brain, most of the time)
- Tinker Toys (Danny Hillis, 1975)
- Quantum Computing
- DNA computing
- Universal Computer

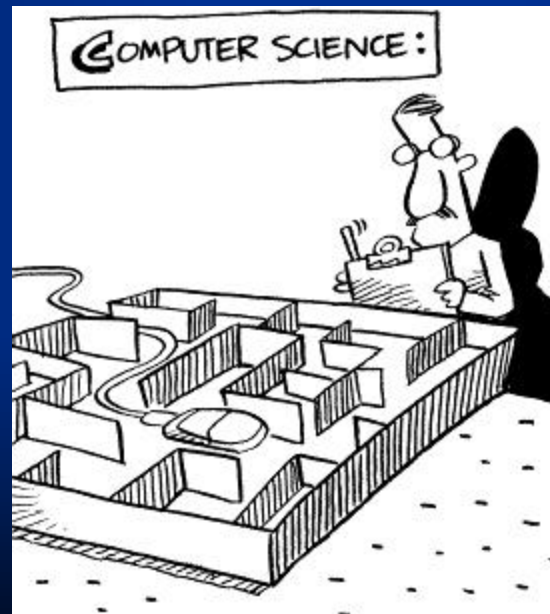
“All processes, whether they are produced by human effort or occur spontaneously in nature, can be viewed as computation”.
Wolfram, A New Kind of Science

Given that so many disciplines are developing software as a tool, what is the role of the software engineer?

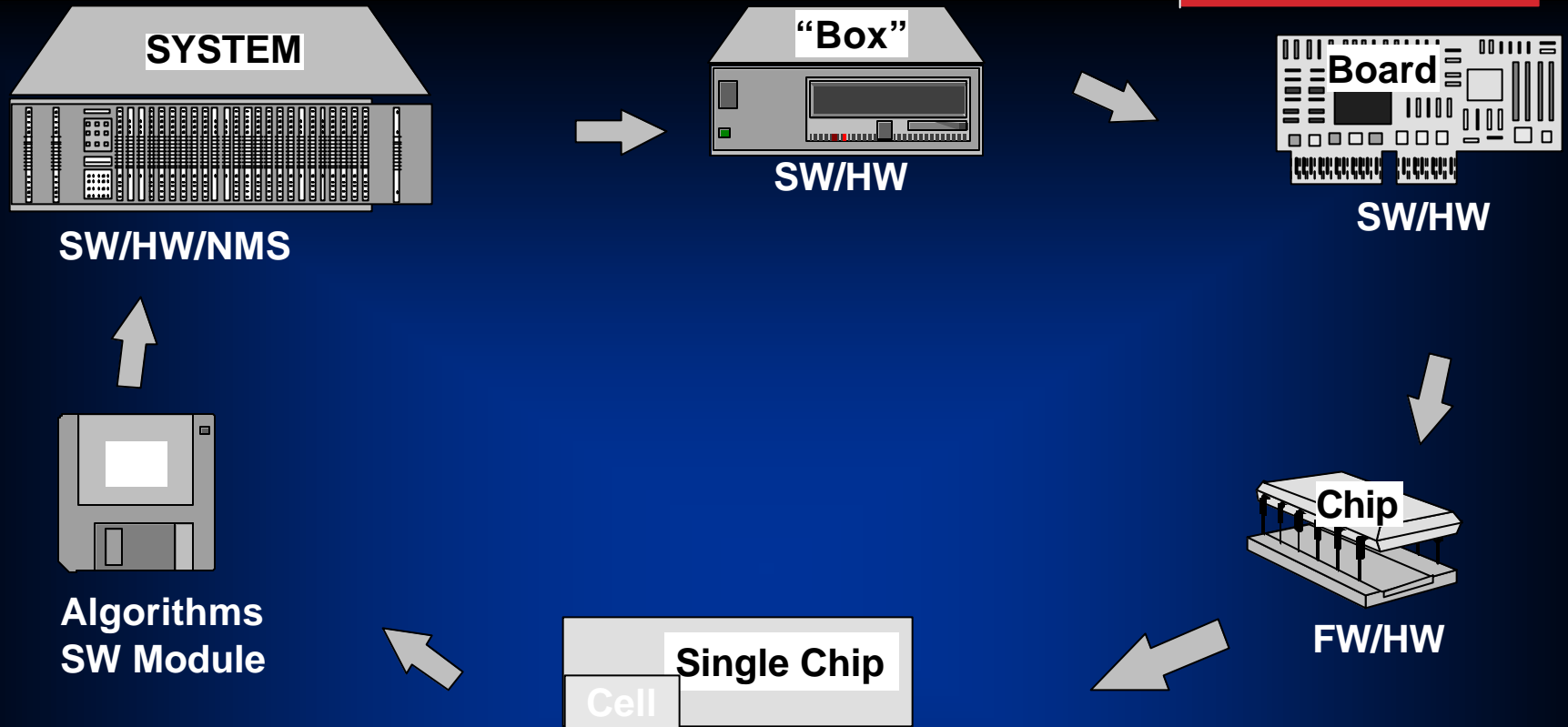
- **Software Engineer:** works primarily in the problem domain. Work with users, design & architect systems, perform analysis, solve technical problems, and oversee the implementation. Requires breadth of expertise across multiple disciplines and technologies
- **Developer/Programmer:** works primarily in the solution domain. Focus is on implementation. Often has a specialty skill.



- Focus primarily on theory, research, and invention.
 - In academia, often perform basic research
 - In industry, typically perform applied research

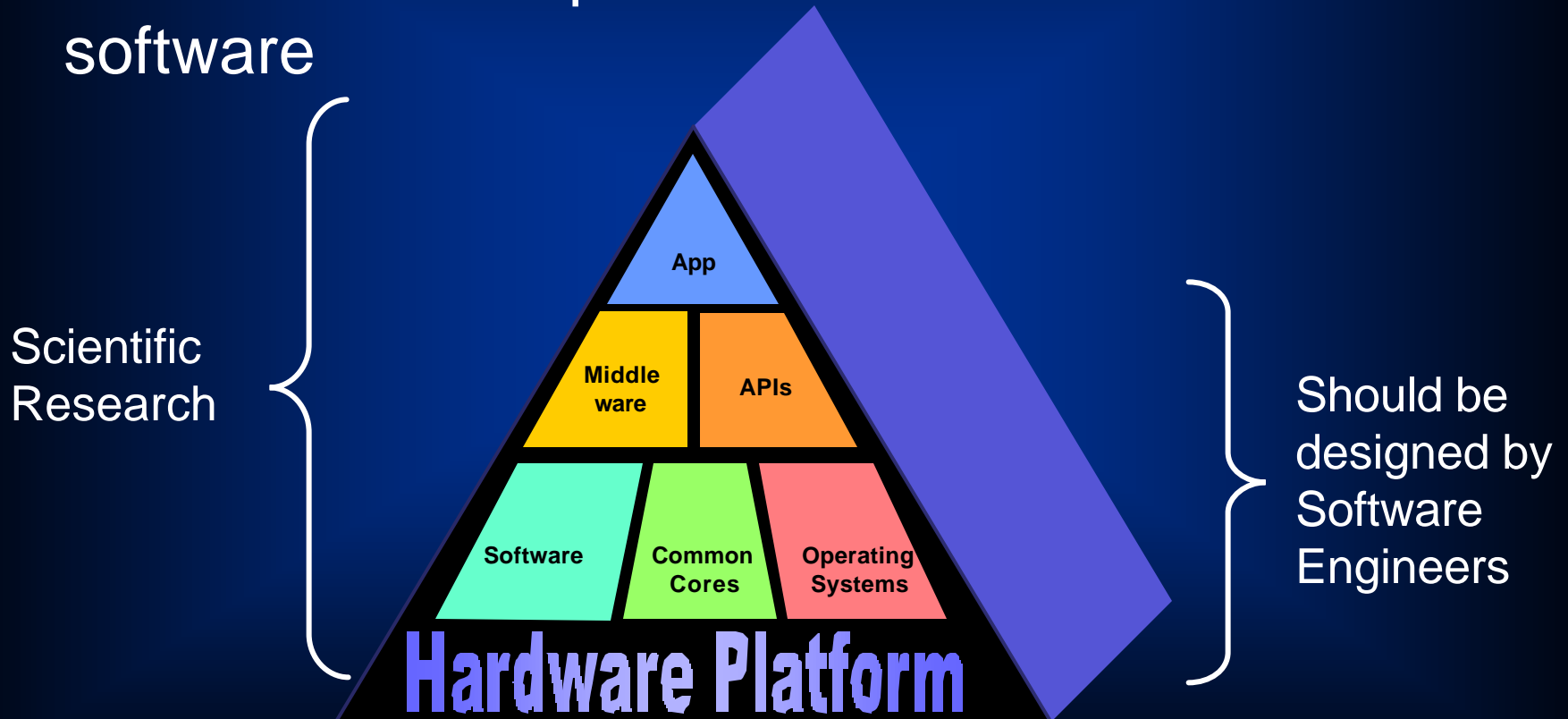


Traditional Technology Cycle

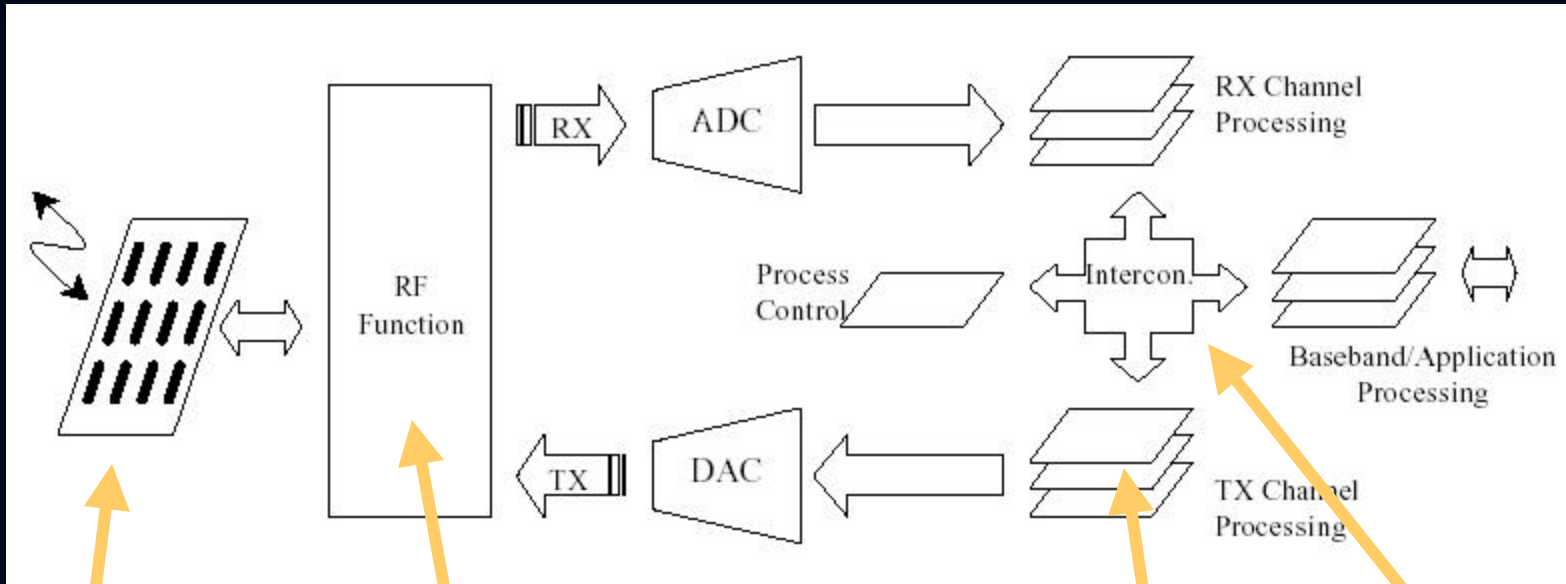


Digitization, Moore's Law, broadband, IP and technology cycle driving convergence

- Functionality and Flexibility from software
- Convergence on common platforms
- Differentiation of products and services from software



Example: Software Defined Radios



Programmable RF
Modules

Embedded DSP
algorithms

Open Architecture
Interconnect

Intelligent
Antenna



Higher technical complexity

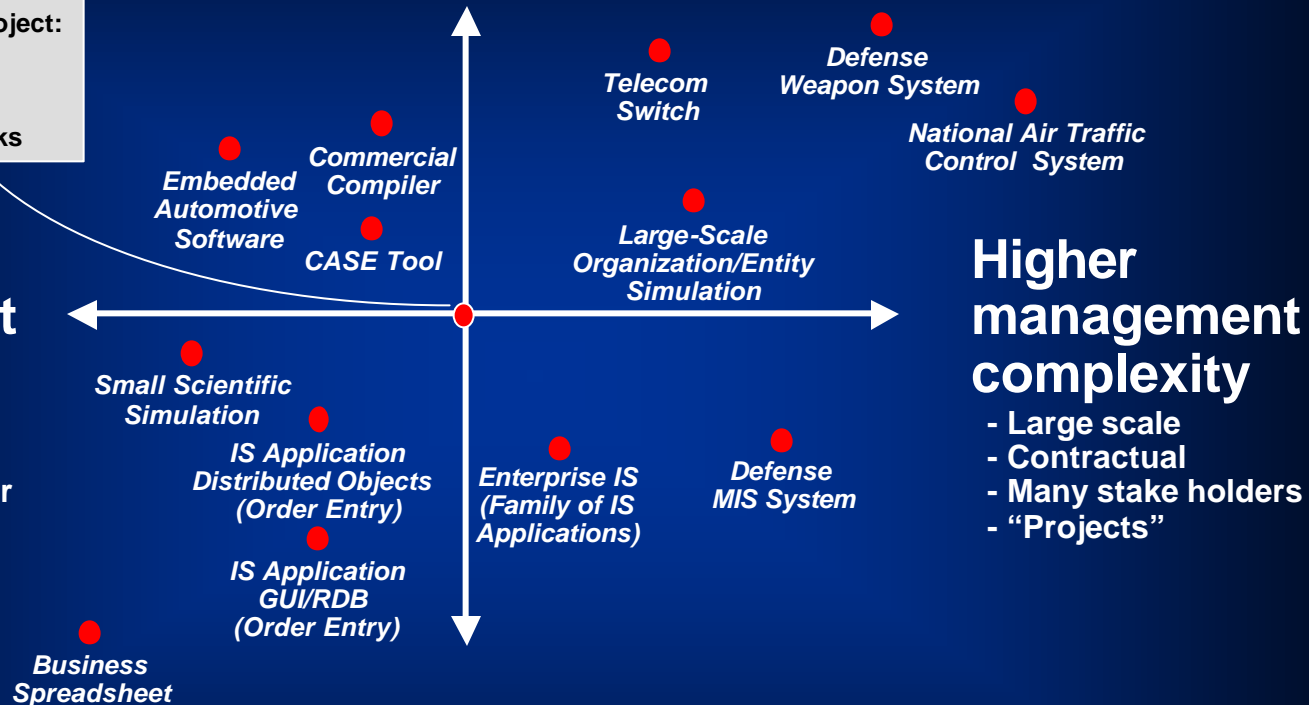
- Embedded, real-time, distributed, fault-tolerant
- Custom, unprecedented, architecture reengineering
- High performance

An average software project:

- 5-10 people
- 10-15 month duration
- 3-5 external interfaces
- Some unknowns & risks

Lower Management complexity

- Small scale
- Informal
- Single stakeholder
- "Products"



Higher management complexity

- Large scale
- Contractual
- Many stake holders
- "Projects"

Lower Technical complexity

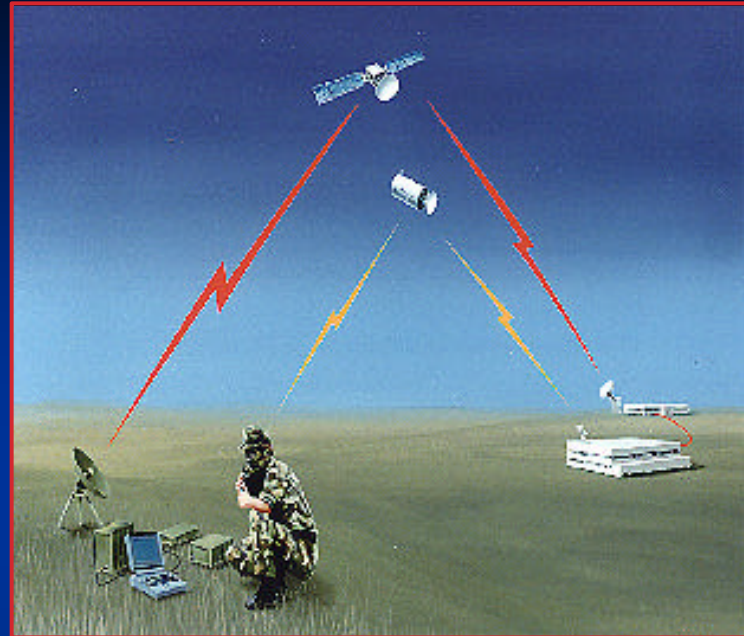
- Mostly 4GL, or component-based
- Application reengineering
- Interactive performance

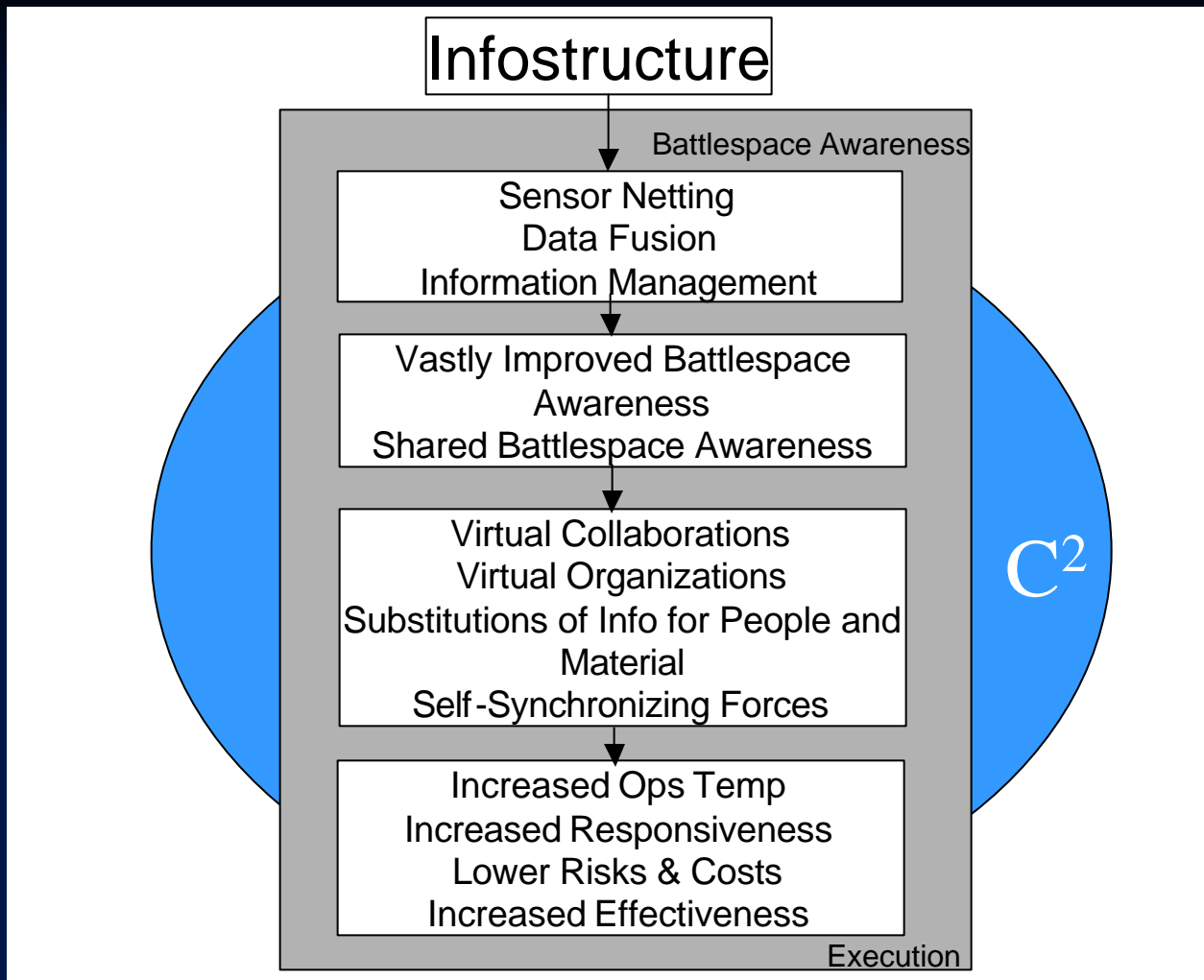
Military Transformation

HARRIS

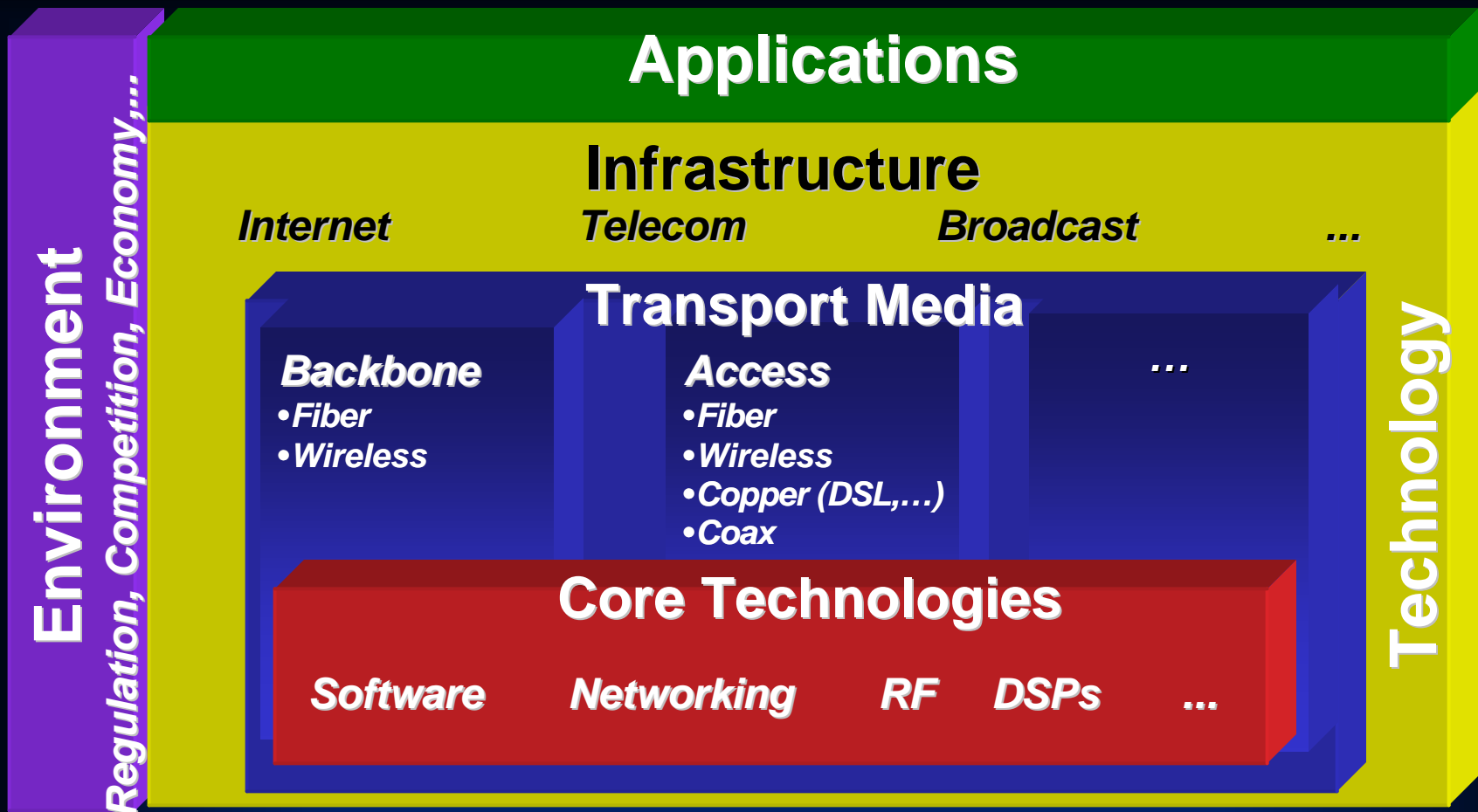


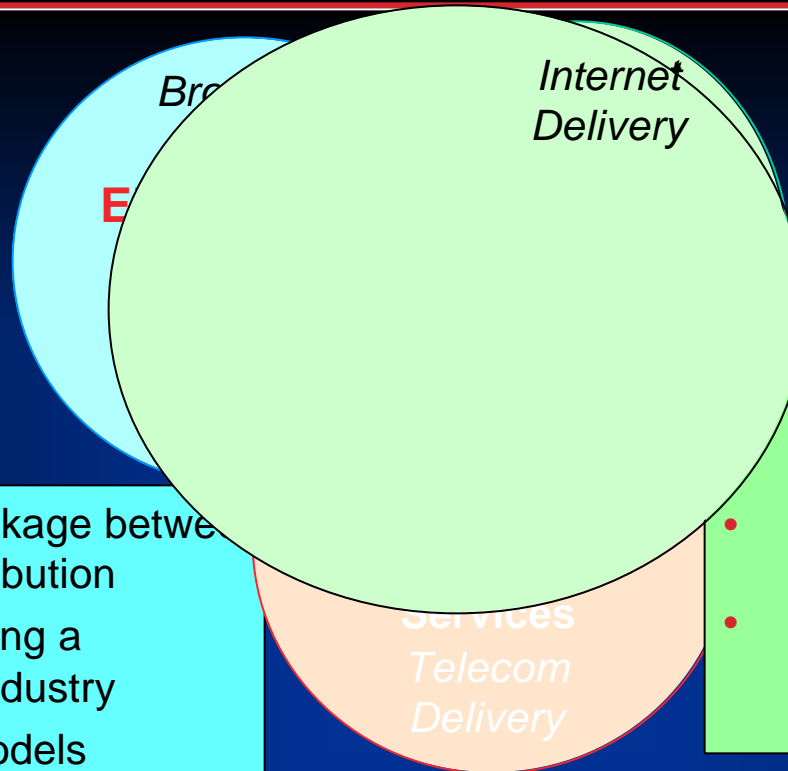
- Any where
- Any time
- Any body
- Every where
- Space to Mud
- Whitehouse to Foxhole





Network Centric Warfare, 2nd ed., D. Alberts, J. Garktka, F. Stein, 1999





- Historic strong linkage between content and distribution
- Digitization creating a discontinuity in industry
- New business models emerging
 - Content generation/distribution
 - Role of local broadcaster
 - New uses of spectrum
- A regulatory environment encouraging competition

The Internet increasingly becoming "the network"

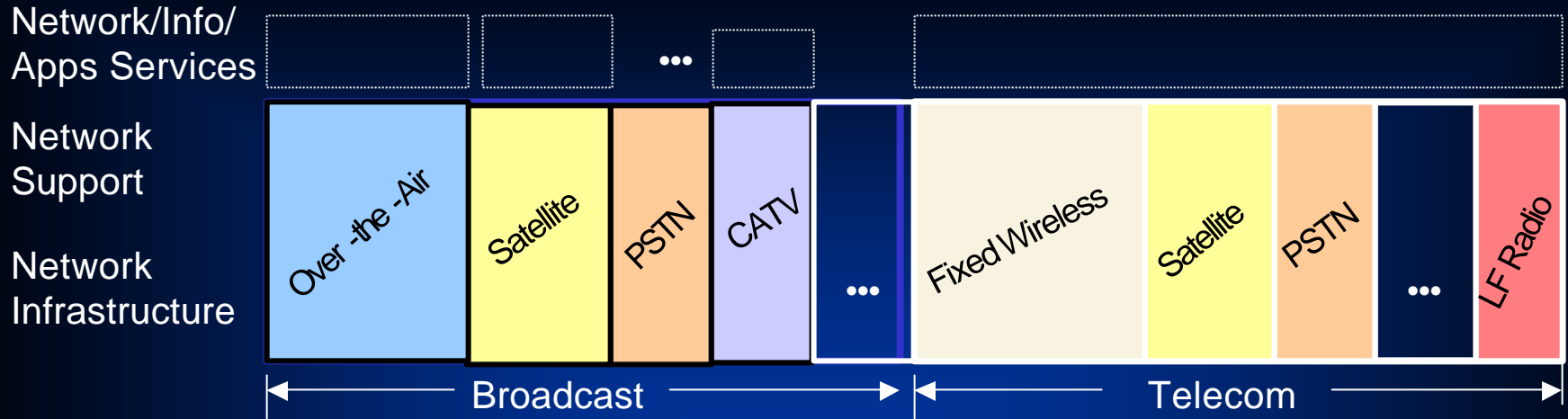
- All applications/content types
- Other networks become subnetworks
- Emerging architecture to support convergence
- Network infrastructure driven by content distribution paradigm

- Major impact of global deregulation/privatization
 - xSPs, layers of value chain becoming distinct
- Driven by new revenue-generating services
 - Explosive data growth
- Network evolving to support all applications and content types: voice, data, video, ...
 - Data-centric, packet, wiloitics/multiple access, ...

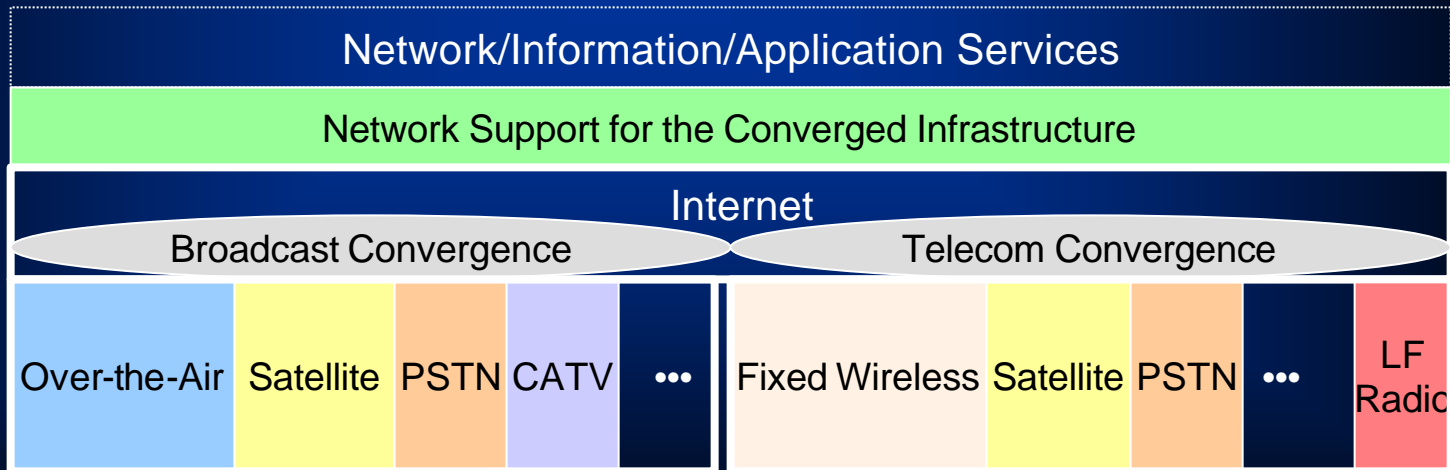
Communications Transformation



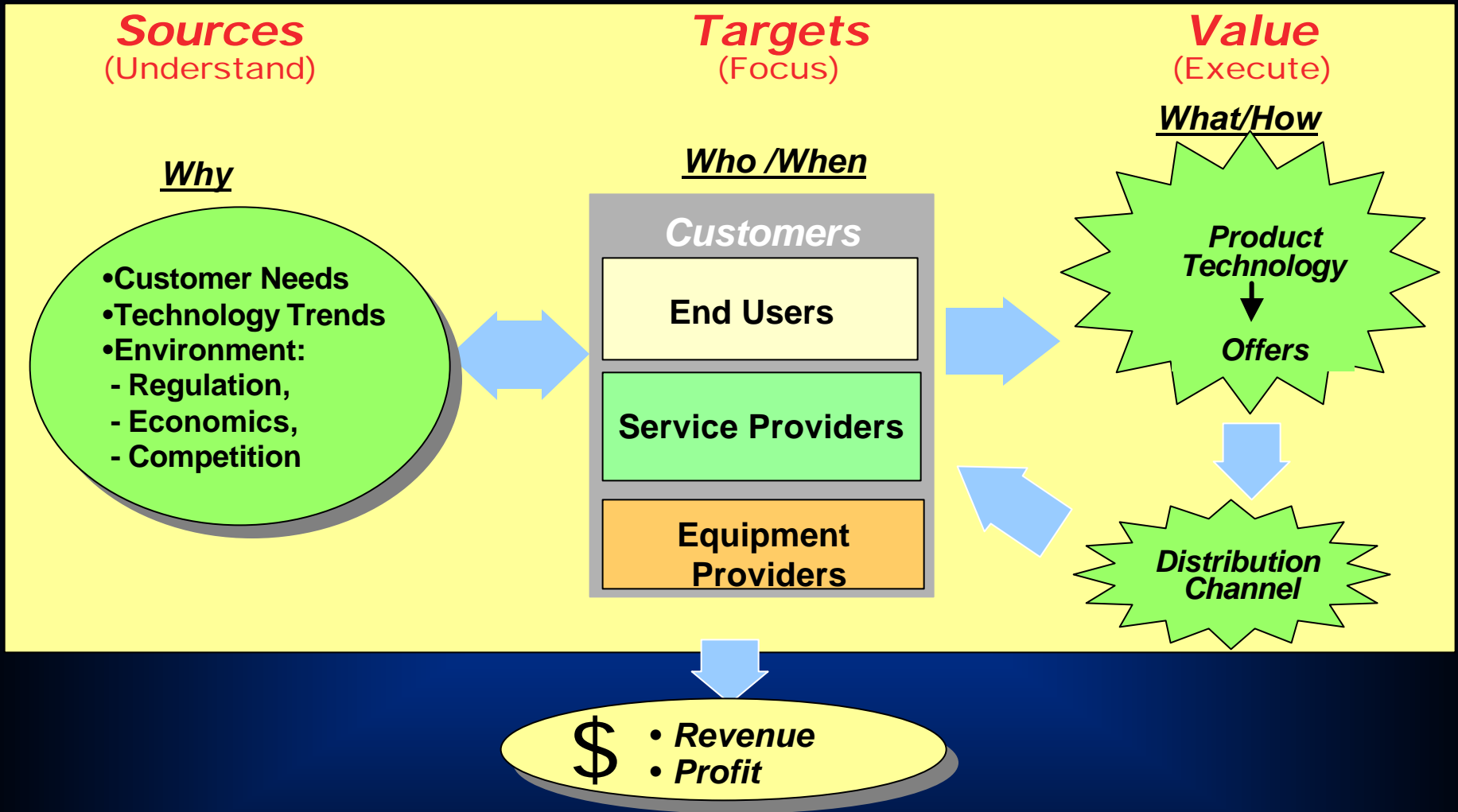
Today



Going Forward

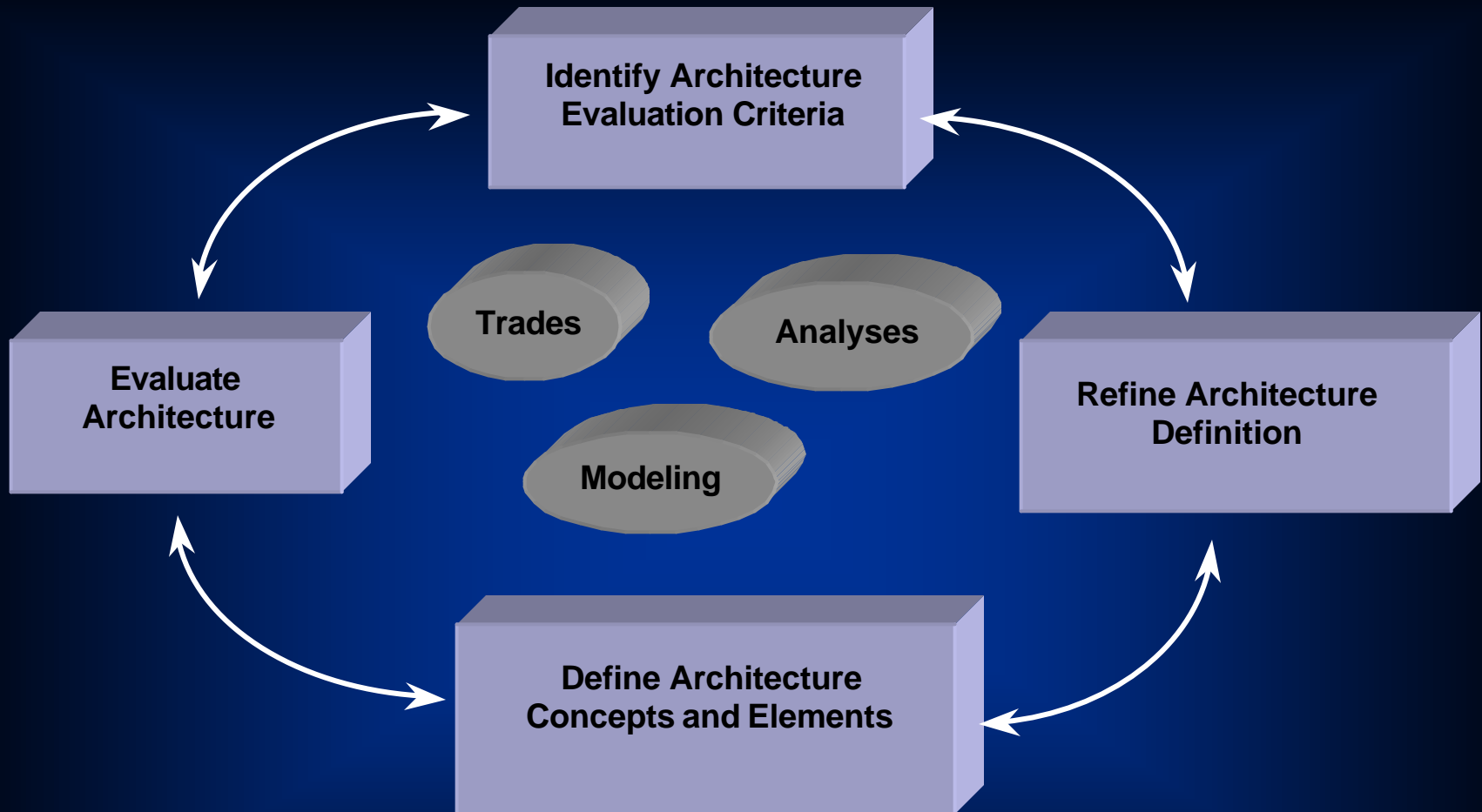


Value Creation Model

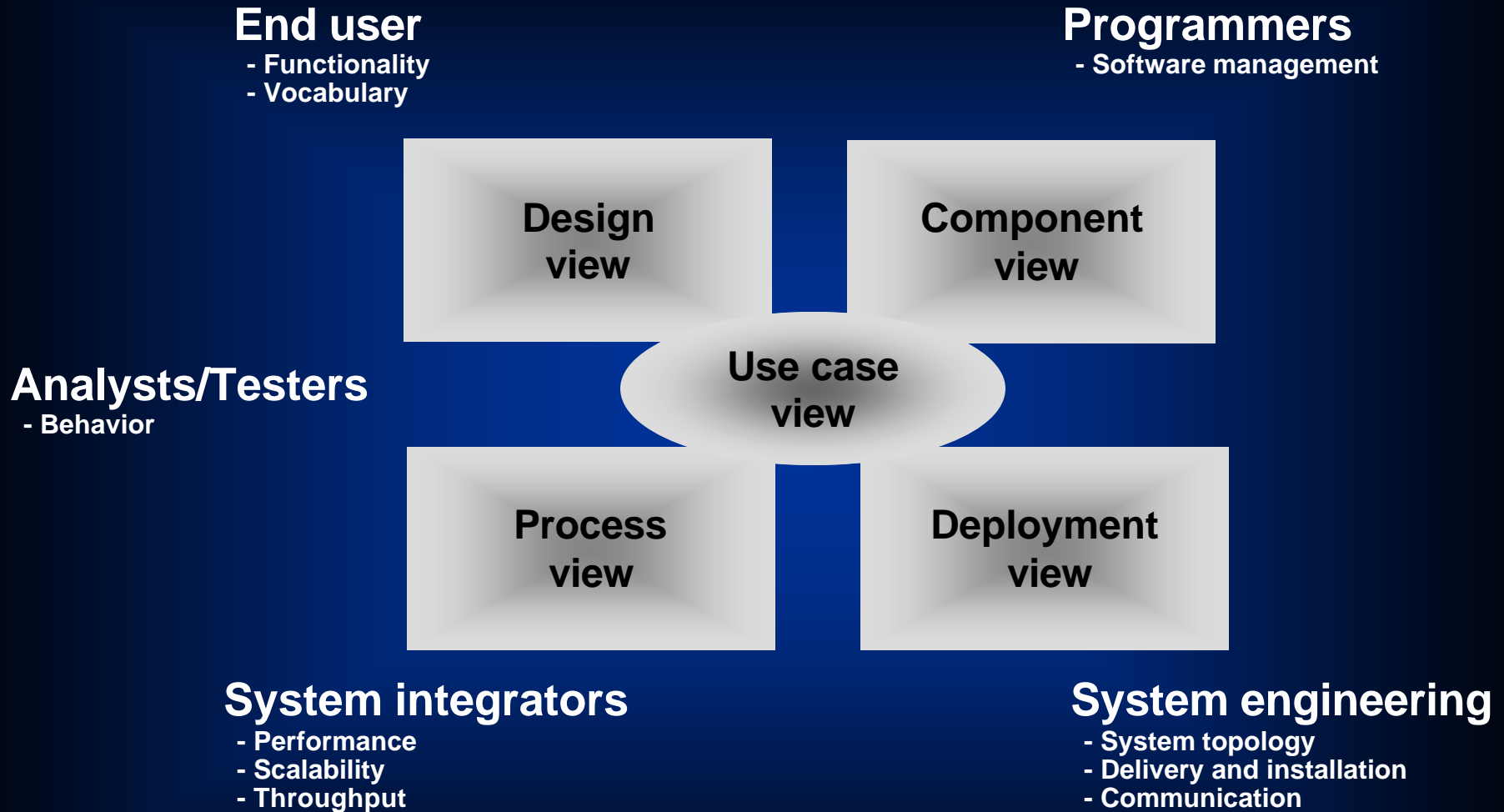


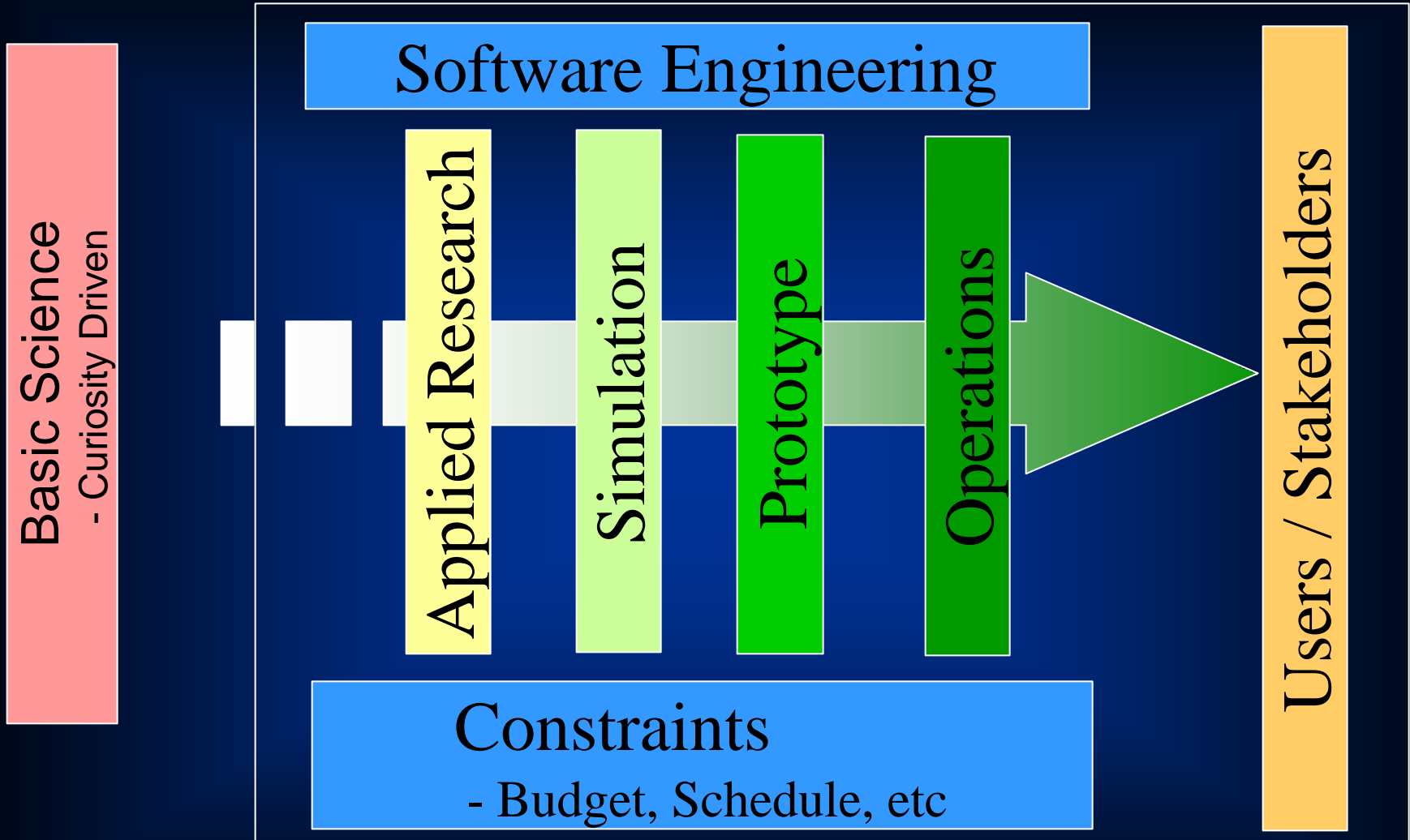
Software Engineers support the lifecycle

System Architecture Process



The 4+1 view model





- When building a house, many specialists might be involved. There might be new research applied. But there must be an architect and an engineer to pull it together.



- Applying sound software engineering processes is critical for large, life-critical systems
- Also important even for small projects, and for projects early in the research phases
 - These projects have a tendency to morph into large systems, or become essential elements of other systems
 - Do it right the first time. Use a software engineer.
- At Harris, our processes are applied to projects ranging from \$200K IR&Ds, to \$3.5B contracts.
 - Common framework

1. Finding and fixing a software problem after delivery costs 100 times more than finding and fixing the problem in early design phases.
2. **You can compress software development schedules 25% of nominal, but no more.**
3. For every \$1 you spend on development, you will spend \$2 on maintenance.
4. Software development and maintenance costs are primarily a function of No. of SLOC.
5. **Variations between people account for the biggest differences in software productivity.**
6. The overall ratio of (SW Costs/HW costs) is still growing 1955: 15/85, 1985: 85/15.
7. **Only about 15% of software development effort is devoted to programming.**
8. Software systems and products typically cost 3 times as much per SLOC than individual software programs. Software-system products (i.e., system of systems) cost 9 times as much.
9. **Walkthroughs catch 60% of the errors.**
10. **80% of the contribution comes from 20% of the contributors.**

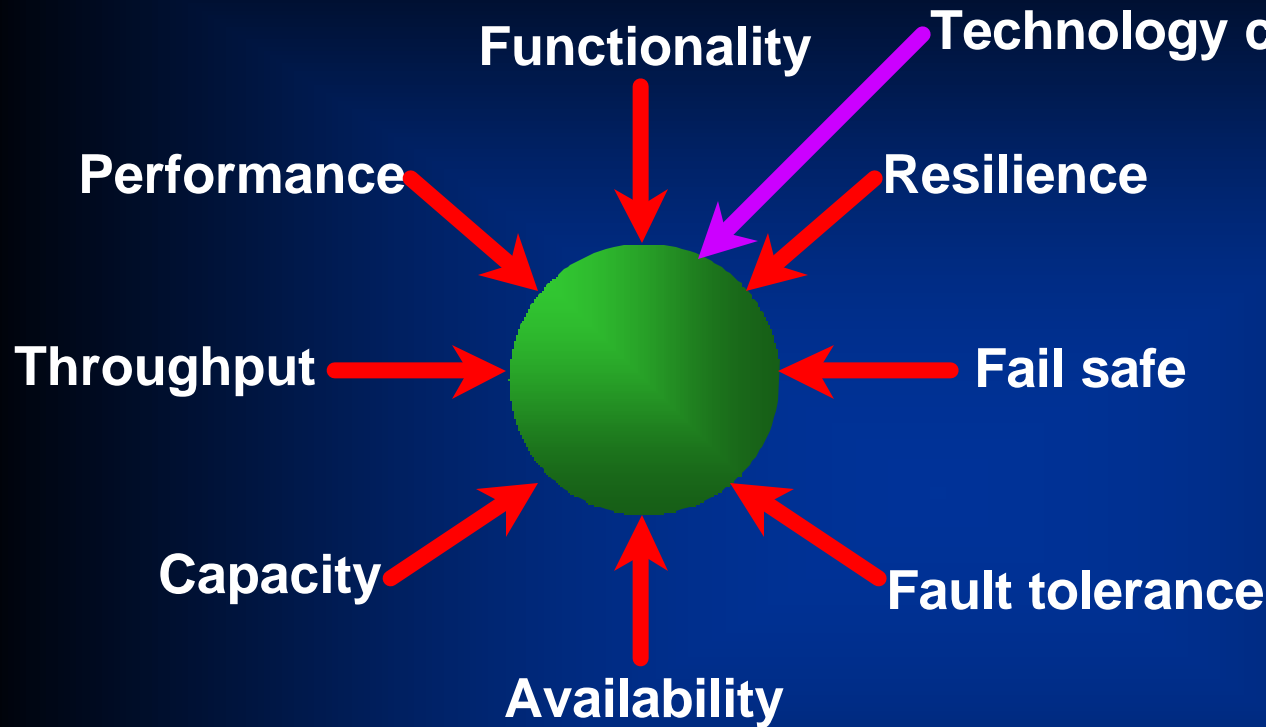
Royce, W. E. *Software Project Management: A Unified Framework*. Addison-Wesley Longman, 1998

- Several important dimensions:
 - 80% of the engineering is consumed by 20% of the requirements
 - 80% of the development cost is consumed by 20% of the components
 - 80% of the errors are caused by 20% of the components
 - 80% of development phase scrap and rework is caused by 20% of the errors
 - 80% of the resource consumption (execution time, disk space, memory) is consumed by 20% of the components
 - 80% of the engineering gets accomplished by 20% of the tools
 - 80% of the progress is made by 20% of the people
- Can you identify the 20%? A Software Engineer should!

- The IT industry has focused too much on the latest & greatest technology
- Not enough focus on ease of use, especially for non-engineering users
- Needs to be part of the engineering process
 - Intuitive
 - Simple administration
 - Secure, trusted
- The best user interface is no user interface. The best technology is invisible technology

*There are only 10 types of people in the world:
Those that understand binary, and those who
don't.*

- Distributed, ad-hoc processing and networks are exceptionally complex
- Very difficult to design, debug, simulate
- Hard to predict the “real world” behavior and usages
- Reliability, maintainability, and security problematic
- Unintended side effects
- Few software engineers are prepared to meet this challenge



Differences

- No moving parts
- New materials can be created
- Physics can be changed

Avoiding failure


- Separation of concerns
- Semantic consistency
- Distribution of responsibilities

Have an architecture that makes sense before you write 3.5 million lines of code.

- Patrick Naughton

Taming the Complexity – Maturity Models



Maturity Level	Focus	Process Areas (PAs)	
5 Optimizing	<i>Continuous Process Improvement</i>	Organizational Innovation and Deployment Causal Analysis and Resolution	Quality Productivity  Risk Rework
4 Quantitatively Managed	<i>Quantitative Management</i>	Organizational Process Performance Quantitative Project Management	
3 Defined	<i>Process Standardization</i>	Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus Organizational Process Definition Organizational Training Integrated Project Management Risk Management Decision Analysis and Resolution	
2 Managed	<i>Basic Project Management</i>	Requirements Management Project Planning Project Monitoring and Control Supplier Agreement Management Measurement and Analysis Process and Product Quality Assurance Configuration Management	
1 Initial			

- The services infrastructure is essential
- Support massive scale, continuous real-time
- Includes the datacenters

- Customers driving requirement for ubiquitous, device independent services
 - From computers
 - From PDAs
 - From cellphones
 - From vehicles
 - From watches
 - From sensors

- Software is everywhere, driving everything
 - Life critical systems
 - Economic critical systems
 - The value discriminator
- Increasingly networked and complex
- Used as a tool for every form of science
- The role and need for the software engineer is clear, but insufficiently used and inadequately taught
- Software engineering can help transition research to mission success